NAME

socket — Open a TCP network connection

SYNOPSIS

socket ?*options*? *host port*
socket -server *command* ?*options*? *port*

DESCRIPTION

This command opens a network socket and returns a channel identifier that may be used in future invocations of commands like **read**, **puts** and **flush**. At present only the TCP network protocol is supported over IPv4 and IPv6; future releases may include support for additional protocols. The **socket** command may be used to open either the client or server side of a connection, depending on whether the **-server** switch is specified.

Note that the default encoding for *all* sockets is the system encoding, as returned by **encoding system**. Most of the time, you will need to use **chan configure** to alter this to something else, such as *utf-8* (ideal for communicating with other Tcl processes) or *iso8859-1* (useful for many network protocols, especially the older ones).

CLIENT SOCKETS

If the **-server** option is not specified, then the client side of a connection is opened and the command returns a channel identifier that can be used for both reading and writing. *Port* and *host* specify a port to connect to; there must be a server accepting connections on this port. *Port* is an integer port number (or service name, where supported and understood by the host operating system) and *host* is either a domain-style name such as **www.tcl.tk** or a numerical IPv4 or IPv6 address such as **127.0.0.1** or **2001:DB8::1**. Use *localhost* to refer to the host on which the command is invoked.

The following options may also be present before *host* to specify additional information about the connection:

-myaddr *addr*
: *Addr* gives the domain-style name or numerical IP address of the client-side network interface to use for the connection. This option may be useful if the client machine has multiple network interfaces. If the option is omitted then the client-side interface will be chosen by the system software.

-myport *port*
: *Port* specifies an integer port number (or service name, where supported and understood by the host operating system) to use for the client's side of the connection. If this option is omitted, the client's port number will be chosen at random by the system software.

-async
: This option will cause the client socket to be connected asynchronously. This means that the socket will be created immediately but may not yet be connected to the server, when the call to **socket** returns.

  When a **gets** or **flush** is done on the socket before the connection attempt succeeds or fails, if the socket is in blocking mode, the operation will wait until the connection is completed or fails. If the socket is in nonblocking mode and a **gets** or **flush** is done on the socket before the connection attempt succeeds or fails, the operation returns immediately and **fblocked** on the socket returns 1. Synchronous client sockets may be switched (after they have connected) to operating in asynchronous mode using:

  ```
  chan configure chan -blocking 0
  ```

  See the **chan configure** command for more details.

  The Tcl event loop should be running while an asynchronous connection is in progress, because it may have to do several connection attempts in the background. Running the event loop also allows you to set up a writable channel event on the socket to get notified when the asynchronous connection has succeeded or failed. See the **vwait** and the **chan** commands for more details on the event loop and channel events.

  The **chan configure** option **-connecting** may be used to check if the connect is still running. To verify a successful connect, the option **-error** may be checked when **-connecting** returned 0.

  Operation without the event queue requires at the moment calls to **chan configure** to advance the internal state machine.

SERVER SOCKETS

If the **-server** option is specified then the new socket will be a server that listens on the given *port* (either an integer or a service name, where supported and understood by the host operating system; if *port* is zero, the operating system will allocate a free port to the server socket which may be discovered by using **chan configure** to read the **-sockname** option). If the host supports both, IPv4 and IPv6, the socket will listen on both address families. Tcl will automatically accept connections to the given port. For each connection Tcl will create a new channel that may be used to communicate with the client. Tcl then invokes *command* (properly a command prefix list, see the **EXAMPLES** below) with three additional arguments: the name of the new channel, the address, in network address notation, of the client's host, and the client's port number.

The following additional option may also be specified before *port*:

-myaddr *addr*
: *Addr* gives the domain-style name or numerical IP address of the server-side network interface to use for the connection. This option may be useful if the server machine has multiple network interfaces. If the option is omitted then the server socket is bound to the wildcard address so that it can accept connections from any interface. If *addr* is a domain name that resolves to multiple IP addresses that are available on the local machine, the socket will listen on all of them.

Server channels cannot be used for input or output; their sole use is to accept new client connections. The channels created for each incoming client connection are opened for input and output. Closing the server channel shuts down the server so that no new connections will be accepted; however, existing connections will be unaffected.

Server sockets depend on the Tcl event mechanism to find out when new connections are opened. If the application does not enter the event loop, for example by invoking the **vwait** command or calling the C procedure **Tcl_DoOneEvent**, then no connections will be accepted.

If *port* is specified as zero, the operating system will allocate an unused port for use as a server socket. The port number actually allocated may be retrieved from the created server socket using the **chan configure** command to retrieve the **-sockname** option as described below.

CONFIGURATION OPTIONS

The **chan configure** command can be used to query several readonly configuration options for socket channels:

-error
: This option gets the current error status of the given socket. This is useful when you need to determine if an asynchronous connect operation succeeded. If there was an error, the error message is returned. If there was no error, an empty string is returned.

  Note that the error status is reset by the read operation; this mimics the underlying getsockopt(SO_ERROR) call.

-sockname
: For client sockets (including the channels that get created when a client connects to a server socket) this option returns a list of three elements, the address, the host name and the port number for the socket. If the host name cannot be computed, the second element is identical to the address, the first element of the list.

  For server sockets this option returns a list of a multiple of three elements each group of which have the same meaning as described above. The list contains more than one group when the server socket was created without **-myaddr** or with the argument to **-myaddr** being a domain name that resolves multiple IP addresses that are local to the invoking host.

-peername
: This option is not supported by server sockets. For client and accepted sockets, this option returns a list of three elements; these are the address, the host name and the port to which the peer socket is connected or bound. If the host name cannot be computed, the second element of the list is identical to the address, its first element.

-connecting
: This option is not supported by server sockets. For client sockets, this option returns 1 if an asyncroneous connect is still in progress, 0 otherwise.

EXAMPLES

Here is a very simple time server:

```
proc Server {startTime channel clientaddr clientport} {
    puts "Connection from $clientaddr registered"
    set now [clock seconds]
    puts $channel [clock format $now]
    puts $channel "[expr {$now - $startTime}] since start"
    close $channel
}

socket -server [list Server [clock seconds]] 9900
vwait forever
```

And here is the corresponding client to talk to the server and extract some information:

```
set server localhost
set sockChan [socket $server 9900]
gets $sockChan line1
gets $sockChan line2
close $sockChan
puts "The time on $server is $line1"
puts "That is [lindex $line2 0]s since the server started"
```

HISTORY

Support for IPv6 was added in Tcl 8.6.

SEE ALSO

chan, **flush**, **open**, **read**

KEYWORDS

asynchronous I/O, bind, channel, connection, domain name, host, network address, socket, tcp